Authenticast: An Adaptive Protocol for High-Performance, Secure Network Applications *

Phyllis A. Schneck and Karsten Schwan

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
phyllis, schwan@cc.gatech.edu

1 Introduction

1.1 Motivation

A primary obstacle in the path to successful commercial Internet utilization is the lack of adequate security. Although some Internet commerce applications have been developed, they do not provide sufficiently strong security for a trusted transfer of private data over a public medium. The very essence of “strong security” is the notion that the security mechanism employed to protect data cannot be reversed in a sufficiently short time to allow use or alteration of those data by an unauthorized party. Therefore, data protection mechanisms for strong security are required to be complex, and they thus have a high computation overhead which detracts from overall application performance. In the interest of performance, security procedures are often omitted. If Internet commerce applications are to succeed, they cannot compromise performance or security. The inherent tradeoffs in realizing these two services comprise the challenge we face in providing them.

Our work addresses the balancing of tradeoffs between security and performance. After all, of what use is Internet security if the delays imposed are so inconvenient that the customer might prefer to walk his money to the bank instead of waiting for the secure transfer on-line? We present Authenticast, an adaptive, user-level authenticated transmission protocol to facilitate a

*This work was funded by DARPA, Aasert grant DAAH04-96-1-0209
balance between these tradeoffs to enable the existence of certain high-performance applications with sufficient security to be executed over the Internet. For example, many commercial applications currently rely on a secure connection. However, this does not address the scenario in which an intruder adds data to the wire to be received and potentially played out. Authenticast provides continuous authentication capability to authenticate a user-determined percentage of all packets and thus decreases the likelihood of added packets reaching playout.

Authenticast also addresses the fact that security algorithms differ in complexity of computations, number and nature of parameters, keys required, and key length [6]. Thus, performance overheads differ in magnitude and location (e.g., overloading sender vs. receiver) depending on what security algorithms are employed with which parameters. For example, Authenticast makes it possible to maintain connection(s) secured by keys of different lengths (discussed in Section 4.7.3), providing a choice of algorithm and key length, as well as the dynamic run-time ability to alter that choice.

Further, Authenticast can protect against “timing attacks,” [7] in which some cryptosystems can be broken by “carefully measuring the amount of time required to perform private key operations.” [7] In dynamically adjusting parameters that affect transmission delay, Authenticast masks the time allotted to security processing and thus may increase the difficulty of obtaining security processing time measurements.

We note that this work does not necessarily target “classified” applications. Our work is intended for those applications that could make use of a public medium, given an opportunity to specify a security level range. ¹

Since Authenticast is software-based, the protocol can be easily provided to a partner host in a transaction. Software also provides the flexibility to support several different cryptographic algorithms, easily replace algorithms as they become outdated, and keep pace with rapidly changing standards. An often-noted advantage to hardware use is improved performance, however, it has

¹Certain applications, such as a video or audio transmission, may contain sections that require different levels of security based on content. Variation in securityLevel allows hosts to exploit those durations that require less security overhead.
been shown that this improvement may not be significant for all cryptographic algorithms [12].

1.2 Contributions

Our solution comprises the following contributions:

- We present the addition of the security allocation parameter, `securityLevel`, to the Quality of Service (QoS) specification, and provide a pre-execution system profiling technique to better determine whether or not the existing system configuration can provide the user-requested security while still satisfying the other QoS requests. This profiling complements existing procedures in the spirit of the connection-oriented paradigm, in which the possibility of granting a requested level of QoS is determined before a connection is granted.

- We introduce the concept of the `Security Thermostat` to depict Authenticast’s dynamic runtime modification of `securityLevel` based on user requirements. The Security Thermostat is an abstraction of the capability that Authenticast provides a user to set the `securityLevel` parameter at runtime, and modify this parameter throughout execution within a given range. The range allows the application to fluctuate and therefore conserve resources for higher performance if needed.

- We present the design and implementation of the Authenticast protocol to interface with the “Thermostat” concept and facilitate parameter changes during application execution. While we specifically discuss a security thermostat, Authenticast is designed to support dynamic modification of QoS parameters in general. For example, Authenticast works well with the DRRM concept, presented in [13], to modify non-security QoS parameters when needed to maintain a given degree of security in transmission.

We have implemented a prototype of a high performance privacy system [14] with strong public key security to enable strongly-authenticated video transmission. In transmissions which require security operations at the hosts, we show performance improvements when using the security thermostat concept over conventional heuristics.
1.3 Roadmap

The remainder of this paper is outlined as follows: Section 2 presents a brief overview of digital signature technology as well as some definitions. In Section 3, we discuss related work in secure MPEG and general adaptive communications and show where our work expands the current picture. Section 4 is a detailed discussion of the tradeoffs that security presents and the mechanisms by which Authenticast addresses the problem. We present our approach, heuristics, and a sample application. We also present reasoning behind the types of applications that may benefit most from Authenticast, and then discuss the extensions that we are currently pursuing. We follow this with our initial Authenticast implementation and performance gains with the sample application in Section 5. This section also provides our reasoning behind the types of applications that may benefit most from Authenticast. We conclude with Section 6.

2 Background – Digital Signature Technology

The Office of Management and Budget (OMB) has asserted that for electronic commerce and Electronic Data Interchange (EDI), digital signature is the most secure technology [2]. Digital signature authentication comprises both the signing/generating and the verifying of a digital signature. Since our implementation of Authenticast employs digital signatures to utilize strong authentication, this section is intended to provide some background on digital signatures and related terms. Encryption and authentication serve different security purposes, and are often employed together. Due to the nature of the computations required, authentication tends to produce more of a load imbalance between sender and receiver, where encryption has more balanced resource requirements. To highlight the effects of load imbalance, we focus on authentication in our implementation of Authenticast.

We establish the following working definitions:

- Encryption

  Encryption is the sufficient distortion of a given data set to minimize its usability in the event
of unauthorized access. For instance, encrypted, unauthenticated data may be accessed by any party that happens to have the decryption function.

- Authentication

*Authentication* is the act of proving the true identity of a host. Digital signature authentication provides a method by which the sender of a data stream and the integrity of those data can be proven to the receiver or even to a third party, since the identity of the sending host can be proven mathematically. Authentication is used to prevent unauthorized access to data, because the accessing party may be asked to first identify himself and verify authorization before any data are sent. Authentication does not provide data privacy or confidentiality.

The Authenticast protocol provides the flexibility necessary for applications to vary authentication algorithms and parameters such as keys and key length. According to Bruce Schneier, “A good cryptographic system strikes a balance between what is possible and what is acceptable,” [14] and the Authenticast protocol can achieve this balance.

3 Related Work

Numerous systems, protocols, and user applications are currently being enhanced to provide greater security and thus increased flexibility for systems and end-users.

SECMP [11], has some parallels with Authenticast in that it offers varied levels of security for encrypted MPEG. SECMPEG includes the capability to encrypt only the most important and significant data, in the spirit of better performance. However, we are addressing not only when and how to apply security but also the general issue of asymmetric end host processing loads. In addition, the SECMPEG security levels are based on the types of MPEG frames are encrypted. Although we are currently investigating selective authentication based on frame type, we emphasize the provision of dynamic user and application-driven varied security levels that can be changed as desired during application execution.
Varied levels of security are also employed in the MPEG player described in [9]. [9] looks also at the issue of security vs. performance, yet focuses again on encryption where we emphasize authentication. As was done in [11], [9] employs information, such as frame type, to select frames to encrypt. Extensions of our work focus on authentication using similar selection tactics to identify frames to be verified. However, our decision processes in choosing particular types of encoded frames to sign or authenticate is dependent not only on overall desired level of security, but also on the security algorithm being used, the current system configuration, and often on the overall MPEG encoding pattern.

[13] presents middleware for creating and dynamically renegotiating connections between an application and a network. We assume this architecture in order to benefit from application and user information during application execution. We address how the ability to create a selective security level in MPEG authentication can be implemented using such a “manager.” Specifically, security algorithms, levels, and key management could be added to this manager as further dimensions of change in a given connection based on input from the application.

4 Using the QoS Specification to Balance Security and Performance

4.1 The Problem is Real

The Authenticast protocol targets the difficulties encountered in executing an application between hosts with different processing capabilities. For example, Figure 1 shows the effect that the addition of another connection, thus increased load at each host, can have an overall information retention. Both graphs are based on the transmission of approximately 656K bytes of data per connection, including signature and control information, representing about two minutes’ worth of video feed, or a brief teleconference. In both graphs, the dips represent ranges in which incoming packets were received but not able to be processed due to the load on the receiving host processor. These stream plots are to illustrate the points on the stream, and the frequency with which, packets are lost at the receiving host.
The upper graph shows the 12% packet loss in processing a single transmission at full verification level. The lower figure depicts the packets lost in processing two simultaneous connections. This shows two streams, and the second stream was plotted with a vertical offset of 130 so that it could be more easily discerned from its companion connection on the same plot. The upper stream had a packet loss of 28% and the lower stream 38%. Both graphs were created using 100% verification (worst-case scenario). Every incoming, signed packet was verified. When another connection is added, the packet loss more than doubles, demonstrating the damage caused by the lagging receiver. Further, the single connection does not experience any data loss until approximately 110 packets have been sent. The load of two connections leads to data loss much sooner.

![Data Loss with 100 % Verification -- Single Connection](image)

![Data Loss with 100 % Verification -- 2 Connections](image)

Figure 1: Example scenario: Effect of additional load on receiver processing capability

### 4.2 The Authenticast Protocol

Figure 2, shows a case in which information retention varies directly with rate control (send delay) and indirectly with security level. The $y$ – axis represents the number of packets that are dropped at the receiver. For the line with the points labeled with *, the $x$ – axis represents ten millisecond increments of delay between actual packets send events. Thus, we see that as this delay increases, packet drops decrease. For the line with the points labeled with @, the $x$-axis represents the percentage/10 of packets that were required to be authenticated at the receiver before being processed.
This shows the increase in information loss as this verification percentage (level of security) is increased. These data are based on the repeated transmission an MPEG movie clipping using the values for sending rate and securityLevel shown. Each transmission consisted of approximately 11.4 Mb of data in 2500 transmissions of packets sized 4592 bytes.

![Diagram](image)

Figure 2: Information retention with * = increased inter-packet send delay and @ = increased security

Note that the packet losses depicted in Figure 2 do not represent a loss in transit. These are packets that are dropped at the destination host, because the cycles at that host are consumed by the security processing for other packets. By the time the receiver is able to go to the network interface and get the next packets, some have already been lost. Authenticast detects this due to a gap greater than 1 in the Authenticast-specific sequence number (discussed in detail in Section 4.7.1 between the last packet processed and the next one taken from the network interface. The ATM signalling on top of which we have implemented Authenticast, guarantees ordered arrivals [5]. Therefore, detecting such a gap in sequence numbers at the receiver is a clear indication that the packets whose sequence numbers were not received are gone. This would invoke a reaction to decrease either sending rate or securityLevel as needed to improve information retention at the receiver.

Figure 2 also portrays the contrasting effects of sending rate and securityLevel on overall
information retention. Clearly making the sending rate too high would hamper performance, and security taken too low may not satisfy user requirements. Further, if too great an average sending rate is permitted at the network level, then that increases the amount by which Authenticast must control send rate at the application level in order to prevent information arriving at the receiver faster than its required security measures can be processed.

Working with application behavior as affected by security is a key contribution of Authenticast. The Authenticast protocol is designed to adapt, as would a thermostat, to the changing needs of secure applications, based on the inclusion of `securityLevel` as part of the QoS Specification. Authenticast takes in current information during the execution of an application pertaining to current performance parameters and user requirements. This input can actually originate from all possible avenues, including the user, the application, or some third party proxy. For example, if a host augments its resources, this may trigger performance changes recognized by Authenticast.

Authenticast has three main aspects to its operation: 1) the use of the QoS Specification to provide information on an application’s security requirements, 2) the mapping of the QoS requirements run-time decisions and incorporation of user-initiated changes in QoS during application execution using a DRRM, and 3) the adaptive characteristics, allowing the protocol itself to initiate parameter changes based on a decision heuristic.

4.3 Realization of QoS

QoS changes, whether initiated by the user or by Authenticast, can affect performance at both the sender and receiver. Depending on which host is involved, these items may be handled differently. For instance, when using a security algorithm that consumes more resources at the receiving host, a load at the sender might actually help reduce the gap in processing between sender and receiver if the receiver is heavily loaded with authentication processing. In contrast, an increased load at the receiver would widen this gap, and should definitely alarm Authenticast to react by adjusting existing QoS parameters, namely sending rate or `securityLevel`.
4.4 Mapping QoS Parameters

When Authenticast reacts to changes, it applies a mapping of the requested change to QoS parameters. First, the QoS parameters must be mapped from the formal user specification to some form understood by Authenticast. For example, when a connection is established, Authenticast reads the QoS contract parameters [5] and determines a maximum loss rate. Though this is a transmission loss rate, Authenticast can use this specification to formulate an information retention rate pertaining to the end hosts, and then ensure that this is maintained along with any additional security parameters. This is followed by the continuous process of ensuring that these parameters are met. We note that Authenticast must still adhere to the original contract made with the ATM network interface [5] at the network level, and that any changes to this contract must be effected by making changes to the network connection, potentially using a DRRM.

The following is a list of possible Authenticast QoS-mapped reactions to the situation in which the receiver has a security-born burden that is degrading overall application performance:

- Reduce the sending rate
- Offer user decreased security level within specified range
- Renegotiate a secure connection
- Halt execution

If the establishment of the connection itself was not authenticated, another possibility is to use a middleware resource manager such as the DRRM to renegotiate a new connection in which the end hosts have had to prove their identity. In this case, the connection would not be established if host identities could not be verified. Once this is accomplished, packets could be transmitted without the overhead of signing and verifying individual packets. Granted, this is certainly far less secure, but, in cases where the load at a host is too great for the application to be usable with strong security, and the secure connection is within a tolerable security range, this is a viable solution.
Finally, Authenticast offers the option to abort the transaction and conduct it at a later time, when the receiver is not as loaded. In either case, no data that are authenticated in a method below the user-specified minimum will be used or displayed.

4.5 Dynamic System Adaptability

![Diagram](image)

Figure 3: Components of end-to-end delay $EEDelay$

As shown in Figure 3, at any given point in application execution, we quantify the end-to-end delay, $EEDelay$, for a packet or frame $F$ (our implementation transmits data on a per-frame basis) with the the sum of delay incurred by send rate, protocol processing at sender and receiver, and network delay. For clarity, our implementation equates the terms packet and frame.

We note that cost factors may also be included in the end-to-end expression for frame playout. If required, Authenticast heuristics employ a cost-benefit analysis to determine appropriate levels for security, rate control, bandwidth allocation, and other dynamic parameters. The decision to incorporate cost information is left to the application and/or user.

As variance in $TransmissionDelay$ is not within our control, we focus on $SendDelay$ and the latencies created by security protocol processing. The majority of these latencies are caused by the signing and verifying operations, emphasized by the asterisks in Figure 3. The relative amount of delay caused by each depends on the security algorithm and key sizes used. Authenticast adapts to minimize this delay and thus overall $EEDelay$ based on configuration and security parameters.
4.6 Pre-profiling of system configuration

Since `securityLevel` is not included in the conventional QoS specification, current connection establishment in ATM does not look at the effects of a request for this parameter when the QoS contract is negotiated. Authenticcast augments current procedure and provides pre-profiling, another level of negotiation – at user level.

Before establishing a connection, Authenticcast looks at the `securityLevel` requested compared to performance bounds specified. By looking at values of QoS parameters, such as `securityLevel` and end-to-end delay, compared to a performance metric, such as packets lost at the receiving host, Authenticcast produces a generalization of whether or not the given performance can be realized with the given QoS, resources, and level security requested. In our initial implementation, send rate is quantified by millisecond delays between sends, and `securityLevel` represents a percentage of packets that are verified before processing at the receiver.

Figure 2 represents sample plots. If the current `securityLevel` request is infeasible with the other QoS parameters, the connection is closed and the contract is renegotiated with QoS to accommodate the requested security. This helps users establish rate and `securityLevel` ranges that can be maintained with the system involved, and it also prevents secure applications from starting execution only to be halted due to lack of resources. For example, we acknowledge that there exist several applications in which `securityLevel` must never be reduced. In those cases, Authenticcast may be of the greatest benefit to interactive applications, which have natural pauses that will allow for the far greater decrease in sending rate that will be needed to compensate for the inability to decrease `securityLevel`. The needed send rate is estimated in the pre-profiling step and utilized by Authenticcast in connection establishment (Section 5.3).
4.7 Authenticcast ADAPT Decision Heuristics

4.7.1 Information included in packet header

Authenticcast decision heuristics rely on specific control information included with a packet. The header of every Authenticcast packet contains an Authenticcast sequence number, Security algorithm ID, Security type, securityLevel range, send rate range, current send queue size, current send rate, current securityLevel, and the digital signature of the sender. This information is specific to Authenticcast, and is independent of any similar fields, such as sequence numbering, created at lower layers.

It is important to note that the Authenticcast header makes up less than 3% of the overall packet size. The benefits gained from having this information included in packets outweighs any space overheads incurred. The bulk of this header is the 80-byte digital signature itself.

4.7.2 ADAPT-SS: Single Stream, Dynamic Intelligent Modification of Rate and securityLevel

Authenticcast dynamic reactions are defined and invoked by one or more decision heuristics within the protocol. ADAPT-SS is the ADAPT decision heuristic utilized in our current implementation of Authenticcast. It is designed to be somewhat predictive, in that it relies heavily on information about system load and behavior trends, and its decisions are sufficiently conservative in order to prevent irreparable performance degradation due to a delayed reaction to user or system changes.

Figure 4 shows the decision flow ADAPT-SS. We make the following assumptions:

- All computations of $EEDelay$ are done at the receiver.
- Initial rate control and security levels are user-specified.
- Values of rate control and securityLevel modification increments and decrements are user-specified.
- ADAPT-SS maintains the highest possible sending rate and securityLevel while still enforcing the user-specified rate and security ranges.

4.7.3 Extensions and work in progress

An obvious extension to ADAPT-SS would be to inject more information into the decision of the amount by which send rate and securityLevel are altered. For send rate modification, instead only allowing modification amounts to be user-specified, extensions to ADAPT-SS include an option in which the Sender computes EEDelay at specified points for different rates, given the current parameters for protocol processing and transmission, and determines the best possible sending rate to maintain the user-specified ranges.

For securityLevel, our initial implementation utilizes the simplistic method of authenticating
a certain percentage of received frames, as was seen in Figure 2. However, tailoring the selection and frequency of authentication to specific traits of the application, such as [9] describes for encryption, may prove more strategic and improve overall performance. We are currently investigating more sensitive ways to balance security and performance. We are specifically studying the effects of utilizing more application-specific security alteration heuristics for authentication. Further, we intend to tailor the way in which we delimit different security levels so that they are more meaningful to those communities requiring security and potential guarantees.

In addition, for this initial implementation, we have provided Authenticast with the public and private keys as needed. We are looking to create more clever key management schemes such as addressing key lookup, achieving interoperability with existing key infrastructures, addressing necessary privacy issues in key generation [6] and adding key length to the security part of the QoS specification. We anticipate exploiting the mathematical properties of keys in conjunction with these efforts to improve the speed and benefits of obtaining and maintaining multiple secure connections that may each require different keys. For example, algorithms such as Elliptical Curve Cryptosystems may prove more efficient for some applications [10], yet they require longer key lengths. We are extending our heuristics to address issues in the overhead of storage and retrieval of these keys and in the maintenance of connections with various key sizes.

5 Implementation

5.1 System Configuration

Our implementation of Authenticast is built in user space on top of ATM (AAL5) [5]. We chose ATM for the prototype Authenticast implementation, because it allows us to reserve network resources and complements our efforts to accurately estimate host resource availability. However, Authenticast heuristics are designed to be compatible with any underlying communications medium. The current hosts are two SGI Indy machines; one acts as a sender, and the other as the receiver. This is a connection-oriented paradigm, thus a QoS contract is negotiated once before a connection is established and maintained for the life of the connection. Rate control and securityLevel
can be altered, because Authenticast manages these parameters above the ATM communication protocol.

5.2 Security

Our implementation of secure MPEG using the Authenticast protocol uses existing DSA secure hash algorithms in the Sign and Verify operations, COTS software from the Digital Signature Engine from CygnaCom Solutions, Inc. The sender signs encoded MPEG data on a per-frame basis to ensure maximum flexibility as to which frames the receiver chooses to verify. The key lengths are 128 bytes and 40 bytes for the public and private key respectively.

5.3 Connection Establishment

Authenticast augments the standard ATM connection establishment procedure. Using the ATM User-Network Interface (UNI) signalling [5] for the send and receive operations at the host interfaces, we establish a connection between sender and receiver. The parameters used in this establishment are the standard ATM QoS parameters [5]. In a regular ATM connection, the connection establishment process would end here, and the connection would be granted. However, an Authenticast connection requires additional negotiation.

If the network determines that a connection can be negotiated based on the standard QoS contract that was requested, we then execute the Authenticast pre-profiling step. If we can grant the requested range of security while also keeping the sending delay within the requested range without violating any of the basic ATM QoS parameters, then Authenticast accepts the connection. If not, then the user is given the option to alter the securityLevel range or send delay request to comply better with the provisions of the ATM QoS contract. If the user cannot produce modifications that will work successfully with the ATM QoS, then Authenticast refuses to grant the connection.
5.4 Adaptive, Secure Transmission

Since authentication with DSA algorithms imposes a greater load on the receiver than on the sender, signing all packets actually alleviates some of the symmetry. If other algorithms, such as RSA, are employed instead of DSA, the load balance between hosts may vary, and Authenticast does provide the opportunity for the sender not to bother signing frames that will not be verified.

Based on securityLevel, a subset of the received frames are verified at the receiver. If the verify function returns an error code, the current frame is not processed further. For example, in this case with a movie stream, unverified frames are not played out. If a tradeoff is necessary, Authenticast prioritizes security over movie quality by default, and provides users the option to alternate securityLevel or other parameters to improve playout. We start with a send delay range of 0-50 milliseconds, and securityLevel range of 50-100% verification. In this initial implementation, securityLevel refers to the percentage of packets whose signature verified at the receiver before data are used.

Here, we follow the heuristic pictured in Figure 4, calculating $\text{EEDelay}$ at the receiver at regular intervals. In addition, Authenticast can detect changes in system load by watching the length of the receive queue. As frames arrive at the receiver, they are queued. Queue length is dynamic, but Authenticast relies on /high and low “water marks” to define when a queue is getting close to extending its maximum or minimum length, as defined by either the user or the protocol. For example, if a queue length is $L$, our high water mark might be at $L - \Delta$. When length approaches the high water mark, Authenticast may react by decreasing sending rate or selecting a lower securityLevel on the security thermostat. The value of $\Delta$ is larger for applications that need faster or more predictive reactions to changing loads, to allow more frames to fill the queue and play out without delay while Authenticast adjusts conditions to prevent any performance degradation.

If the receive queue length hovers at the low water mark, this means that securityLevel or sending rate may be increased, thus improving security offered or playout. This water mark method of warning will also signal changes that are the result of changes in transmission speed or security.
5.5 Initial Results

Figure 5: Improvement through dynamic adjustment of percentage of packets verified

Figure 5 demonstrates an 83% improvement in information retention through dynamic modification of level of security. In each plot, the $x$ axis represents an Authenticast sequence number, and the $y$ axis represents whether or not a packet was retained at the receiver. The zero-value indicates a packet was dropped by the receiver before being processed. Both plots depict a transmission of 12.8 Mb, divided into 2781 packets of size 4592 bytes.

Both transmissions have a 50 millisecond delay between sends. The upper plot maintains 100% packet verification at the receiver, and shows the loss of 509 packets at the receiver. The lower plot is an example of a transmission that allowed the specification of a range of securityLevel from 100% down to 50%. In this case, Authenticast’s security thermostat mechanism decreased the securityLevel whenever the receiver became overloaded throughout the transmission. The dynamic heuristic led to only 88 packets being dropped. Clearly, the improved transmission leads to fewer interruptions and thus overall greater application quality and performance. We are currently building on these initial results as described in Section 4.7.3.
6 Conclusions

To repeat a question posed in the Introduction, *of what use is Internet security if the delays imposed are so inconvenient that the customer might prefer to walk his money to the bank instead of waiting for the secure transfer on-line?*

For today’s commercial applications to make successful use of the broad access of public media such as the Internet, they require security and performance. The Authenticast protocol fosters the coexistence of these two components to create the synergy required to satisfy commercial application demands.

By adding `securityLevel` to the existing QoS parameter portfolio and introducing Authenticast to work with the concept of the security thermostat, we enable a methodology that is mindful of application performance and user convenience while maintaining a specified level of security. Authenticast allows applications that require a certain range of security to still benefit from the accessibility provided by widespread public communications media such as the Internet.

7 Acknowledgements

The authors would like to thank CygnaCom Solutions, Inc., http://www.cygna.com.com, for the use of their digital signature engine. Our security algorithms are from that engine, a CygnaCom Solutions product. This company also furnished sample keys and exponents to enable performance-testing of these algorithms with our configuration. We would especially like to thank Dr. Santosh Chokhani for his expertise and advice on security issues and Mr. Isadore Schoen for his help in incorporating the CygnaCom Solutions digital signature engine into our work.

References


